

Homework Assignment 5

November 8, 2001

Due Date: November 29

Practice Problem

Consider an undirected $2n$ vertex graph G where it is not known which edges are in G . (In other words, you can think of the adjacency matrix as being hidden from the algorithm.) An algorithm works under a model of computation in which it can give any two vertices, and is told if there is an edge between them. The goal of the algorithm is to determine if the graph is connected while asking about the fewest vertex pairs possible. Use the adversary lower bound technique to prove that at least n^2 vertex pairs must be tested (in the worst case) in order for any algorithm to determine if G is connected.

Hint: The adversary may find it convenient to partition the vertices into two groups of n completely connected vertices. While it is possible to prove a better lower bound, you need not do so.

The solution can be found on the course web page under handouts.

Core Problems

1. (10 pts) Prove that $n - 1 + \min(k - 1, n - k)$ is a lower bound on the number of comparisons needed to find the k th largest element.

Hint: Think about how to modify the lower bound given in class for finding the median of n elements

2. (10 pts) Consider the problem of determining whether an undirected graph with n vertices has a path of length at least 2. Assume that the algorithm is restricted to ask questions of the form “Is there an edge between i and j ?”. Give the best adversary argument you can on the number of such questions required to determine if the graph has a path of length 2 or more.
3. (15 pts) Consider the problem of merging two sorted arrays a_1, a_2, \dots, a_n and b_1, b_2, \dots, b_n . Prove the best lower bound you can on the number of comparisons required.
4. (15 pts) Consider the following simplified version of the game Battleship. For positive integers n and k , there is $kn \times kn$ grid in which a $k \times k$ battleship has been hidden. The algorithm can only learn about the location of the ship by using *probes*. In a probe, the algorithm selects any grid entry (i, j) for $1 \leq i \leq kn$ and $1 \leq j \leq kn$ and is either told “hit” if the ship covers this grid entry or “miss” otherwise. Give the best lower bound you can on the number of probes required by the algorithm to determine the coordinates of the upper-left hand corner of the ship.
5. (10 pts) Consider the “searching for a hole in the fence” problem that was considered in class. You are initially standing at the origin of the real line. The hole is at some unknown

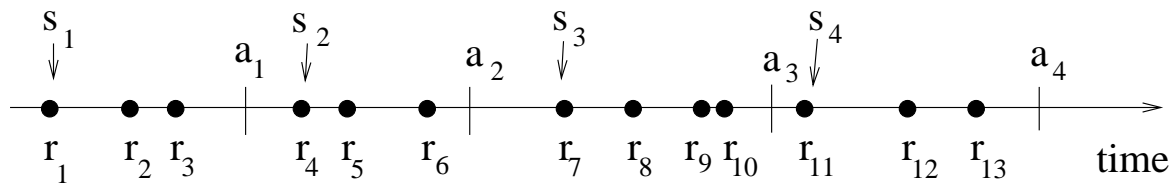
(positive or negative) integer coordinate h . Yu can move left or right at a cost equal to the distance moved, and the game continues until you reach h . The goal is to minimize the distance traveled. In class we gave a deterministic on-line algorithm with a competitive ratio of 9 (which is optimal for deterministic strategies). In this problem you are to describe and analyze a randomized algorithm whose competitive ratio is at most 7.

Advanced Problems (Required only for CS 539T students)

6. (15 pts) Using an adversary argument to show that *any* comparison-based algorithm to find the second smallest of n elements must make at least $n + \lceil \lg n \rceil - 2$ comparisons. (Although I'm not asking you to find it, there is a comparison-based algorithm that makes at most $n + \lceil \lg n \rceil - 2$ comparisons.)

Hint: Partition the input elements into groups in which each group has a minimum element that is known to be less than (or equal to) all elements in its group. Then decide how to answer comparisons in the following three cases: comparing two non-minimum elements, comparing a minimum element to a non-minimum element, and comparing two minimum elements.

7. (15 pts) Consider the following on-line problem. There is a sequence of packets received at times r_1, \dots, r_n where $r_i < r_{i+1}$. The goal is to position ACKS (acknowledgments) within the sequence where each ACK will acknowledge all packets that have arrived since the last ACK. We now define the cost of a solution. Let $A = \langle a_1, \dots, a_k \rangle$ be the times of the ACKS where $1 \leq k \leq n$, $a_i < a_{i+1}$ and $a_k \geq r_n$ (i.e. the last ACK follows the last packet received). For $2 \leq i \leq k$ let s_i be the time of the next packet received after time a_{i-1} and for notational convenience let $s_1 = r_1$. Thus s_i is the time when the earliest packet in the sequence of packets acknowledged by the ACK at time a_i is received. Here is a figure illustrating these definitions.



Then the cost of $A = \langle a_1, \dots, a_k \rangle$ is defined as

$$\text{cost}(A) = ck + \sum_{i=1}^k (a_i - s_i)$$

where c is a constant that corresponds to the cost of an ACK. The goal of the on-line algorithm is to minimize cost A . Note that the algorithm selects k , the number of ACKS to use.

Give a 2-competitive algorithm for this problem.